

Lecture 6: Authentication and Signatures

Lecturer: Nir Bitansky

Scribes: Tamar Bar-Ilan, Yonatan Itay, Michael Kellner

1 Previously on Foundations of Crypto

So far, our focus throughout the course was on *privacy* — starting from OWFs, we developed tools like PRGs and PRFs, which in turn led to secret-key encryption schemes for multiple messages. Today, we will change focus and discuss *authenticity*.

2 The Identification Problem

Let us start by considering a very basic *identification problem*. In this problem, Alice simply wants to prove to Bob that she's indeed Alice and not an impersonator. The adversary now, may do more than just listen in on their conversation — it may use the information it gathers in one conversation to try and impersonate Alice in a later conversation. Can we prevent this?

As before, we assume that both parties have met at least once and have agreed on a shared secret. Perhaps the most naïve solution to the problem is that to prove her identity, Alice would simply send Bob the joint secret key, a *password* if you'd like. This clearly fails as once the adversary figures out this password, it can use it from that point on to impersonate. So perhaps Alice should send the password encrypted? A moment of thought reveals that simply sending the encrypted message also fails — it is subject to pretty much the same *replay attack*.

One-Time Puzzles. A simple solution for the problem is the following. Every time that Alice would like to prove her identity, Bob will generate a fresh new *puzzle* that only she could solve. For instance, using a multi-message encryption scheme, Bob can encrypt a random string under their shared secret key and ask that Alice decrypts it. In fact, if Alice and Bob have a shared clock, we can completely avoid interaction. To identify herself at time t , Alice would need to send $F_{sk}(t)$, where F is a PRF with an n -bit output.¹

Indeed, PRFs essentially give a way of (non-interactively) generating a hard puzzle P_x for any string x in their domain (whose solution is simply $F_{sk}(x)$). Another useful feature is that for any string x , we can delegate the ability to verify the solution for the puzzle for any specific x , without exposing the solution — for instance, we can provide the verifier with $f(F_{sk}(x))$ for a OWF f .

3 Message Authentication Codes

We've sketched how to solve the identification problem, but what if we have a more resourceful adversary, who can hijack the communication line in the middle of the conversation, after Alice had already identified herself. We need a stronger *per-message* authenticity guarantee. This is captured by the notion of *message authentication codes*.

Definition 3.1 (Message Authentication Code (MAC)). *A message authentication code consists of polynomial time algorithms $(Auth, Ver)$ with the following properties.*

- **Syntax:**

¹Last lecture, we've defined PRFs with a single output bit. It is not hard to show that such PRFs imply PRFs with an arbitrary polynomial output.

- $Auth_{sk}(m)$ is a possibly randomized algorithm that takes as input a secret key $sk \in \{0,1\}^n$ and message $m \in \{0,1\}^*$, and outputs a tag t .
- $Ver_{sk}(m,t)$ is a deterministic algorithm that takes as input the secret key sk , message m , and tag t , and outputs bit (where we shall agree that "1" means accept).

- **Correctness:** for any message $m \in \{0,1\}^*$,

$$\Pr[Ver_{sk}(m, Auth_{sk}(m)) = 1] = 1 .$$

- **Unforgeability against Chosen Plaintext Attack:** For any oracle aided n.u. PPT $A = \{A_n\}$ there exists a negligible μ such that

$$\Pr \left[\begin{array}{l} (m^*, t^*) \leftarrow A_n^{Auth_{sk}(\cdot)} \\ m^* \notin Q, Ver_{sk}(m^*, t^*) = 1 \end{array} \right] \leq \mu(n) ,$$

where $sk \leftarrow \{0,1\}^n$ and Q is the set of queries that A_n makes to $Auth_{sk}(\cdot)$.

The above definition is indeed a rather strong form of unforgeability, which guarantees that even if the adversary can see tags on polynomially many arbitrary messages, it cannot generate a valid tag for a new message.

MACs from PRFs. Intuitively, MACs exactly require that each message is associated with a hard-to-solve puzzle that can be solved by anyone that has the secret key, and otherwise is hard to solve (even after the attacker witnessed solutions for other puzzles). PRFs give exactly that.

Claim 3.2. Let F be a PRF, such that for any seed $sk \in \{0,1\}^n$ F_{sk} maps $\{0,1\}^*$ to $\{0,1\}^n$, then the following is a MAC:

- $Auth_{sk}(m) := F_{sk}(m)$.
- $Ver_{sk}(m,t) = 1$ iff $t = F_{sk}(m)$.

Proof Sketch. The idea behind the proof is the following. First, by the pseudorandomness of F , it is sufficient to prove that the scheme is secure when $Auth$ and Ver are given access to a random function $R(\cdot)$. (This is similar to our proof from last week for multi-message security, and common to many other PRF-based proof. Make sure you understand why it is true.)

For a random function, the tag $R(m^*)$ of any message that has not been previously queried is a random and independent value, and thus can be predicted with probability at most 2^{-n} . \square

A Note on the Length of Messages. We've defined PRFs for inputs of length n . As we shall see in the homework, it is possible to extend PRFs for n -bit inputs to inputs of arbitrary length.

4 Signature Schemes

MACs allow us to authenticate peer to peer communication in scenarios where both parties share a secret key. Digital signatures go beyond that allowing to authenticate content in a way that can be publicly verified by anyone. For instance, in *public ledgers* such as bitcoin, users perform transactions of the form "Alice paid \$10 to Bob" and post them to ledger. Here everyone should be able to verify that this transaction was indeed performed by Alice, and thus we'd like Alice to be able to *sign* the transaction in a way that anyone could verify.

Definition 4.1 (Signature Scheme). A signature scheme consists of polynomial-time algorithms $(Gen, Sign, Ver)$ with the following properties.

- **Syntax:**

- $Gen(1^n)$ is a randomized algorithm that takes as input a security parameter n and outputs a secret key sk and a public verification key vk .
- $Sign_{sk}(m)$ is a possibly randomized algorithm that takes as input a secret key $sk \in \{0,1\}^n$ and message $m \in \{0,1\}^*$, and outputs a signature σ .
- $Ver_{vk}(m, \sigma)$ is a deterministic algorithm that takes as input the verification key vk , message m , and signature σ , and outputs bit (where we shall agree that "1" means accept).

- **Correctness:** for any message $m \in \{0,1\}^*$,

$$\Pr [Ver_{vk}(m, Sign_{sk}(m)) = 1 \mid (sk, vk) \leftarrow Gen(1^n)] = 1 .$$

- **Unforgeability against Chosen Plaintext Attack:** For any oracle aided n.u. PPT $A = \{A_n\}$ there exists a negligible μ such that

$$\Pr \left[\begin{array}{l} (m^*, \sigma^*) \leftarrow A_n^{Sign_{sk}(\cdot)}(vk) \\ m^* \notin Q, Ver_{vk}(m^*, \sigma^*) = 1 \end{array} \right] \leq \mu(n) ,$$

where $(sk, vk) \leftarrow Gen(1^n)$ and Q is the set of queries that A_n makes to $Sign_{sk}(\cdot)$.

Signature schemes are the first *public key* primitive we've seen so far. For a long time, it was believed that constructing them inherently requires stronger computational assumptions than those required for secret-key cryptography (OWFs); specifically, assumptions of the type used to construct public-key encryption schemes (we'll hear more about those later on). Surprisingly, it turned out that signature schemes can be constructed from OWFs after all.

Theorem 4.2. [Lam79, GMR84, Gol86, NY89, Rom90] *If there exist OWFs then there exist digital signature schemes.*

We'll now prove a weaker version of this theorem that also relies on *collision-resistant hashing*. We will then explain, only at high level how to remove this extra assumption.

Step 1: One-Time Signatures. We'll start by constructing a weaker object called a *one-time signature* where the adversary can ask for a signature only on a single message of her choice and should then try to forge a signature on a different message.

Definition 4.3 (One-Message Unforgeability). *A signature scheme is one-time unforgeable if for any oracle aided n.u. PPT $A = \{A_n\}$ that makes a single oracle query there exists a negligible μ such that*

$$\Pr \left[\begin{array}{l} (m^*, \sigma^*) \leftarrow A_n^{Sign_{sk}(\cdot)}(vk) \\ m^* \neq q, Ver_{vk}(m^*, \sigma^*) = 1 \end{array} \right] \leq \mu(n) ,$$

where $(sk, vk) \leftarrow Gen(1^n)$ and q is the query that A_n makes to $Sign_{sk}(\cdot)$.

Step 1.1: Fixed Length Messages. Let us first construct a one-time signature scheme for messages of fixed length n . The construction is as follows:

- $sk = \{x_{i,b} : i \in [n], b \in \{0,1\}\}$ where each $x_{i,b} \leftarrow \{0,1\}^n$.
- $vk = \{f(x_{i,b}) : i \in [n], b \in \{0,1\}\}$, where f is a OWF.
- $Sign_{sk}(m) = \{x_{i,m_i} : i \in [n]\}$
- To verify, apply the function and compare to the verification key.

Claim 4.4. *The above is a secure one-time signature.*

Proof Sketch. Let A be an adversary that breaks the scheme with probability $\varepsilon(n)$, we'll construct an adversary B (with polynomially related running time) that inverts f with probability $\varepsilon/2n$.

$B(y)$:

- Samples keys sk, vk as in the actual scheme.
- Samples i, b , and replaces $f(x_{i,b})$ with y , to obtain a new secret key vk' .
- It now emulates $A(vk')$ as follows:
 - When A makes a query m to be signed, if $m_i \neq b$, then B can sign using the secret key sk . If $m_i = b$ it aborts.
 - When A outputs a forgery m^* , $\{x_{j,m_j^*}^* : j \in [n]\}$, if $m_i^* = b$, B outputs $x_{i,m_i^*}^*$ as a preimage for y ; otherwise, it aborts.

We claim that B inverts with probability at least $\varepsilon/2n$. First, note that whenever A produces a valid forgery, then $x_{i,m_i^*}^*$ is a preimage of y . We next analyze the probability that a valid forgery occurs.

Consider an alternative adversary A' for the signature scheme that like B samples i, b at random, and if $m_i = b$ or $m_i^* \neq b$ aborts. Then A' successfully forges a signature with probability $\varepsilon/2n$. Indeed, fix any transcript of the original experiment with respect to A that results in a forgery (such a transcript is fixed by the choice of vk, sk and coins for A). Then this transcript includes a query m and forgery message m^* such that for some j $m_j \neq m_j^*$. The probability that i, b chosen by A' coincide with j, m_j^* is at least $1/2n$. In this case, A' succeeds. Overall, it forges with probability at least $\varepsilon/2n$.

It is left to note that the view of A when emulated by B is distributed identically as the view of A' , and thus a forgery is produced with probability at least $\varepsilon/2n$. \square

Step 1.2: One-Time Signatures for Arbitrary Length. A limitation of the scheme we've just constructed is that it supported messages of fixed size n — the size of the keys for the scheme scaled with n . Our next step will be to obtain one-time schemes where the size of the keys is independent of the length of messages. (Jumping forward, short keys will prove crucial in the next step when extending one-time signatures to many-time signatures.)

For this purpose, we will introduce a very useful cryptographic tool — *collision-resistant hashing*. These are compressing functions where it is hard to find collisions (although they certainly exist).

Definition 4.5 (Collision-Resistant Hash (CRH)). *A collision-resistant hash function is given by a polynomial time algorithm H that given a key $hk \in \{0, 1\}^n$ and input $x \in \{0, 1\}^*$ outputs a hash $H_{hk}(x) \in \{0, 1\}^n$. Furthermore, for any n.u. PPT $A = \{A_n\}_{n \in \mathbb{N}}$ there exists a negligible μ such that for all $n \in \mathbb{N}$,*

$$\Pr \left[\begin{array}{c} H_{hk}(x) = H_{hk}(x') \\ x \neq x' \end{array} \mid \begin{array}{c} hk \leftarrow \{0, 1\}^n \\ (x, x') \leftarrow A_n(hk) \end{array} \right] \leq \mu(n) .$$

Given a collision-resistant function, we can now transform any one-time scheme $(Gen, Sign, Ver)$ for messages of fixed length into a one-time scheme $(Gen', Sign', Ver')$ for messages of arbitrary length:

- $Gen'(1^n)$ samples $(sk, vk) \leftarrow Gen(1^n)$ and $hk \leftarrow \{0, 1\}^n$, and outputs $sk' = (sk, hk), vk' = (vk, hk)$.
- $Sign'_{sk'}(m)$ outputs $Sign_{sk}(H_{hk}(m))$.
- $Ver'_{vk'}(m, \sigma)$ output $Ver_{vk}(H_{hk}(m), \sigma)$.

Claim 4.6. *The above is a secure one-time signature.*

Proof Sketch. Let A be an adversary that breaks the scheme with probability $\varepsilon(n)$, we'll construct an adversary B that finds collisions for a random key hk with probability $\varepsilon - n^{-\omega(1)}$. $B(hk)$ will sample sk, vk by herself, append hk to derive sk', vk' and emulate $A(vk')$. We claim that, except with negligible probability, whenever A makes a query m and produces a forgery on $m^* \neq m$, it is the case that $H_{hk}(m) = H_{hk}(m^*)$, namely it finds a collision. Indeed, if this is not the case then A breaks the underlying one-time signature scheme. \square

Can Collision-Resistant Hash Functions Be Constructed from OWFs? We do not know the answer to this question, and in fact, it can be shown that *standard proof techniques* cannot establish such a reduction. Still collision-resistance is considered a relatively mild assumption and has many candidates. Later on, we'll mention how in the context of signatures we can actually rely on a weaker object that only guarantees *targeted collision resistance*.

Step 2: A Stateful Scheme for Many Messages. We now wish to extend our one-time scheme into a scheme that can resist an arbitrary number of signing queries. As a first step, we'll construct a stateful scheme where at every point t in time, the signer will possess a state $state_t$. The idea here is to generate a *chain of trust* (which is somewhat analogous to the *chain of secrecy* we've seen in previous encryption constructions). At every point in time we will generate a fresh one-time pair of keys, and every time in addition to the message also sign the new verification key. The idea is that if we trust the first verification key in the chain, we'll be able to trust what follows. (Think why to fulfill this strategy it is crucial that the keys in the one-time scheme don't scale with the signed messages.)

We next describe the scheme (a bit less formally than we usually do).

- **State:** After $t \geq 0$ signatures, the state includes:
 - A current one-time signing-key sk_t .
 - A list L_t consisting of triplets $(m_1, vk_1, \sigma_1) \dots, (m_t, vk_t, \sigma_t)$ including previous messages, verification keys and signatures.
- **Signing:** At time $t + 1$, to sign m_{t+1} :
 - Sample a new one-time pair (sk_{t+1}, vk_{t+1}) .
 - Compute a signature $\sigma_{t+1} = \text{Sign}_{sk_t}(m_{t+1}, vk_{t+1})$.
 - Update L_t to obtain L_{t+1} .
 - Send L_{t+1} as the signature.
 - Store sk_{t+1} .
- **Verifying:** given initial verification key vk_0 , message m_{t+1} and signature L_{t+1} , verify the chain starting from vk_0 .

Claim 4.7. *The above is a secure (stateful) signature scheme.*

Proof Sketch. Let A be an adversary that breaks the scheme with probability $\varepsilon(n)$ and at most $t(n)$ queries, we'll construct an adversary B that breaks the one-time scheme with probability $\varepsilon/(t + 1)$. The basic idea is that to forge a signature on a message outside the chain, A must somewhere create a fork in the chain, or extend it, and that's where it breaks the one-time scheme. Our reduction will guess where this happens and simulate the rest.

The adversary $B(vk)$ guesses $i \in \{0, \dots, t\}$. It sets $vk_i = vk$, and samples

$$(sk_0, vk_0), \dots, (sk_{i-1}, vk_{i-1}), (sk_{i+1}, vk_{i+1}), \dots, (sk_{t-1}, vk_{t-1}), (sk_t, vk_t) .$$

It then emulates $A(vk_0)$, it answers queries $1, \dots, i$ using the sampled secret keys, and on the $i + 1$ st query m_{i+1} , it queries its own (one-time) oracle on (m_{i+1}, vk_{i+1}) . (In the case that $i = t$, there is no $i + 1$ st query, and no query is made to the one-time oracle.) From there on, it again uses its own sampled keys. Then, when A produces a forgery L^* for m^* , B checks whether L^* first diverges from L_t previously produced in the $i + 1$ st triplet in the two chains. If not, it aborts. If so, it uses the corresponding message (m_{i+1}^*, vk_{i+1}^*) and one-time signature σ_{i+1}^* as the forgery.

The analysis of the success probability is similar to that in the one-time scheme from OWFs and uses the fact that i is chosen independently of the view of A (until the point it is truncated). \square

Step 3: A Stateless Scheme. The previous scheme had two evident problems. First, it was stateful. Second, the length of the signatures L_t grew with the number of signatures. By now, we already know how to solve both problems using *trees and PRFs*. The idea is to look at a full binary tree with nodes $X = \{x_1 \dots x_i \in \{0, 1\}^i, 0 \leq i \in \mathbb{N}\}$ and associate for each node $x \in X$ a one-time pair (sk_x, vk_x) . Then the signature on message $m \in \{0, 1\}^*$, will include the chain of signatures from the root to the node $m \in X$. Of course that we cannot generate all pairs ahead of time — we will generate them using PRF (to deal with messages of arbitrary length, we'll use a PRF for inputs of arbitrary length).

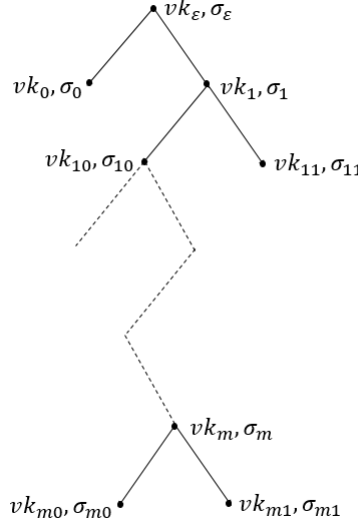


Figure 1: A signature on message m includes verification keys vk_{x_0}, vk_{x_1} and signatures σ_x on the path from the root to m with intermediate nodes x . The keys (sk_x, vk_x) are computed by $Gen(1^n; F_s(x))$ and signatures σ_x by $sign_{sk_x}(vk_{x_0}, vk_{x_1})$.

The Construction: In what follows $(Gen', Sign', Ver')$ is a one-time signature for arbitrary length messages and F is a PRF for arbitrary length inputs.

- $Gen(1^n)$: sample a PRF seed $s \in \{0, 1\}^n$. Compute $(sk_\epsilon, vk_\epsilon) \leftarrow Gen'(1^n; F_s(\epsilon))$. Save s as the secret key sk and publish vk_ϵ as the verification key vk .
- $Sign_{sk}(m)$: For every prefix x of m (including the empty one ϵ):
 - compute $sk_x, vk_x \leftarrow Gen'(1^n; F_s(x))$.
 - compute $\sigma_x = Sign_{sk_x}(vk_{x_0}, vk_{x_1})$.

Output $\{\sigma_x, vk_{x_0}, vk_{x_1}\}_x$.

- $Ver_{vk}(m, \sigma)$: verify all the signatures from the root to the node m .

Note that indeed this scheme is stateless, and the size of the keys is fixed independently of the size of signed messages. The length of the signatures themselves does scale with the length of messages (this can also be dealt with using extra hashing.)

Claim 4.8. *The above scheme is secure assuming the one-time scheme is secure.*

Proof Sketch. First, note that the generator algorithm Gen' makes can be described as an efficient oracle-aided algorithm $Gen'^{F_s(\cdot)}$ that makes black-box calls to the sample PRF. We first note that it suffices to

prove that the scheme is secure when Gen' is given instead of a PRF access to a truly random function R . Indeed, any adversary that breaks the scheme when Gen' has access to a PRF, will break the scheme also when Gen' has access to a random function with the same probability up to a negligible difference. Otherwise such an adversary could be used to efficiently distinguish a PRF oracle from a random function oracle (indeed, the entire signature game can be efficiently emulated).

Next, we show that when Gen' is given a random oracle, the scheme is secure. Let us assume a n.u. PPT adversary $A = \{A_n\}_n$ that breaks the multi-message scheme with probability $\varepsilon(n)$. We will use A to construct an adversary $\{A'_n\}$ that breaks the one-time signature scheme with probability at least $\frac{\varepsilon(n)}{2n \cdot q(n) + 1}$, where $q(n)$ is an upper bound on the number of queries A_n makes. Let $\ell(n) = 2n \cdot q(n) + 1$. Given a verification key vk , A_n will sample $i^* \leftarrow [\ell]$ and construct (sk_i, vk_i) for $i = 1, \dots, \ell$ as follows:

1. If $i \neq i^*$, $(sk_i, vk_i) \leftarrow Gen(1^n)$
2. Otherwise $vk_i = vk$ (and sk_i will not be used).

We simulate A_n with vk_1 . We will assign signing and verification keys to the nodes in the binary tree in the following lazy fashion. When A_n asks for a signature on a message m :

1. For each prefix x of m whose corresponding node has not been assigned keys, assign the next pair of keys on the list (each pair of keys is assigned to at most one node) as vk_{x0}, vk_{x1} , and compute $\sigma_x = \text{Sign}_{sk_x}(vk_{x0}, vk_{x1})$. for every $i \neq i^*$ the signature is computed using the secret keys sampled by A'_n , whereas for i^* , the signature is computed using the (one-time) oracle.
2. Output $\{\sigma_x, vk_{x0}, vk_{x1}\}_x$.

Given A_n 's answer m^* , if its signature $\{\sigma_x^*, vk_{x0}^*, vk_{x1}^*\}_x$ is valid and m^* is a message which hasn't been signed before, then the following is true: There exists a prefix x of m^* where $vk_{x0}^* \neq vk_{x0}$ or $vk_{x1}^* \neq vk_{x1}$. We take the shortest such prefix x , and assume for simplicity that $vk_{x0}^* \neq vk_{x0}$ (the proof for the other case is identical). Hence, if $vk_{i^*} = vk_x$, A'_n outputs a signature σ_x^* on the message (vk_{x0}^*, vk_{x1}^*) .

We first notice that the view of A_n when emulated by A'_n is the same as its view in the experiment for breaking the multi-message scheme. Hence, the probability that A_n will succeed in forging a message is $\varepsilon(n)$. In the case that A_n succeeds, if in addition $vk_{i^*} = vk_x$, then A'_n succeeds in forging a valid signature. Since i^* is sampled at random and independently of the sampled keys (and thus also of A_n 's view), $vk_{i^*} = vk_x$ with probability $\frac{1}{\ell(n)}$. Overall, A'_n will succeed with probability $\frac{\varepsilon(n)}{\ell(n)}$. \square

How to Get Rid of Collision Resistance. To reduce the underlying assumption from CRHs to OWFs, the crucial observation is that a weaker form of collision resistance is sufficient. In this weaker version, we only require that for any (adversarially chosen) input x it is hard to find a sibling x' such that $H_{hk}(x) = H_{hk}(x')$, when hk is chosen at random (after x).

Definition 4.9 (Targeted Collision-Resistant Hash (TCRH)). *A targeted collision-resistant hash function is given by a polynomial time algorithm H that given a key $hk \in \{0, 1\}^n$ and input $x \in \{0, 1\}^*$ outputs a hash $H_{hk}(x) \in \{0, 1\}^n$. Furthermore, for any n.u. PPT $A = \{A_n\}_{n \in \mathbb{N}}$ and any sequence $\{x_n\}_{n \in \mathbb{N}}$ of polynomial-size strings, there exists a negligible μ such that for all $n \in \mathbb{N}$,*

$$\Pr \left[\begin{array}{c} H_{hk}(x_n) = H_{hk}(x') \\ x_n \neq x' \end{array} \mid \begin{array}{c} hk \leftarrow \{0, 1\}^n \\ x' \leftarrow A_n(hk, x_n) \end{array} \right] \leq \mu(n) .$$

Remark: a common definition in the literature allows A_n to sample x . In the non-uniform setting, this is equivalent to the above definition.

Theorem 4.10 ([NY89, Rom90]). *If there exist OWFs, then there exist TCRHs.*

The proof of the above theorem is not trivial and is out of the scope of this course. Let us give some intuition though on why such hash functions are enough for signature schemes. For this, we need to go back to our length extension for one-time signatures. If we simply replace CRHs with TCRHs as is in the construction we've seen it won't work (try to think why). The observation is that in the one-time signature game the adversary first commits to some query message m , and only then gets a signature, which gives us the opportunity to sample a key for the TCRH. Concretely, the new signature would be

$$hk, \text{Sign}_{sk}(hk, H_{hk}(m))$$

(try to think why this works).

References

- [GMR84] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A "paradoxical" solution to the signature problem (abstract). In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, page 467, 1984.
- [Gol86] Oded Goldreich. Two remarks concerning the goldwasser-micali-rivest signature scheme. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 104–110, 1986.
- [Lam79] Leslie Lamport. Constructing digital signatures from a one way function. Technical report, October 1979.
- [NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 33–43, 1989.
- [Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394, 1990.